Rev.1.00
Dec.05.07
REJ10J1738-0100

High-performance Embedded Workshop V.4.04

Release Note

Contents

1. Notes on Using the High-performance Embedded Workshop

This section shows notes on using the High-performance Embedded Workshop.

1.1 Network drive

1.1.1 Time discrepancies between computers

The time kept by a computer differs between computers. The time when a source file or output file is updated depends on the computer on which the file has been saved. Building of a project can be incorrect because of this difference in times kept by computers if a source file or output file is shared via a network. If such a problem occurs, adjust the time between all computers or use [Build -> Build All] to build a project.

1.1.2 Building a project on a network drive

When you build a project on the network an error might occur depending on the condition of the network. For example, a C/C++ compiler might output the following error message.

```
C3019(F) Cannot open source file
```

If such an error occurs, build the project again.

1.1.3 Adding a file to a project

When you add a file on a remote drive on the network via [Project -> Add Files] or else, the High-performance Embedded Workshop might cause an application error depending on the condition of the network. In this case, try to add the file to the project again. If application errors occur many times, consider copying the file to a local drive then adding to the project.

1.2 Comments of the C/C++ language

1.2.1 Syntax coloring of the High-performance Embedded Workshop editor

With the syntax coloring capability of the High-performance Embedded Workshop, comments and keywords in a source program file are colored. But the syntax coloring of the High-performance Embedded Workshop editor does not work correctly depending on the way the source code is written. Disable the syntax coloring if you are not satisfied with the syntax coloring. To disable it, select [Setup -> Options], uncheck “Enable Syntax Coloring” on the [Editor] tab of the [Options] dialog box, and click on the OK button.

Syntax coloring does not work correctly in the following cases:

Case 1. A comment is nested.

e.g. `/* /*  */ */` <- the underlined part is colored as comment

Case 2. “`/*`” or “`*/`” is placed in a string.

e.g.
```
/*
char A[]="*/";      <- the underlined part is colored as comment
*/
```

To prevent this problem, avoid writing * or / characters adjacent to each other (unless such characters are part of a comment) in a C/C++ source file.

1.2.2 File dependencies scanned by the High-performance Embedded Workshop

The High-performance Embedded Workshop scans an include file of a C/C++ source file to show dependencies of source files on the “Projects” tab of the “Workspace” window and to decide whether to build a file or not. Even though the source code is syntactically correct, an error message may appear indicating that an include file has not been correctly recognized as a dependent file. In such cases, (a) build all files by selecting [Build -> Build All] or (b) select the source file in the “Projects” tab of the “Workspace” window and select [Build -> Build (file name)].

Dependencies are not scanned correctly in the following cases:
Case 1. “/*” or “*/” is placed in a string.

e.g.
```
char A[] = “/*”;  <- assumed as beginning of comment incorrectly
#include “file.h” <- this file is not scanned as include file
char B[] = “*/”;  <- assumed as end of comment incorrectly
```

To support preprocessor statements such as #if, #ifdef, and #define by the High-performance Embedded Workshop, go to the [Build] menu to open the Toolchain Option dialog box. On the [Toolchain Option] tab, check [Support dependency scan of preprocessor statement].

Table 1.1 shows the preprocessor statements that are currently supported.

Table 1.1 Preprocessor Statements

Preprocessor Statement	Description
#define	Defines an identifier. When any preprocessor statements include the identifier, this identifier is replaced with the defined character string. e.g. #define NICE_FILE “nice.h” #include NICE_FILE
#undef	Disables replacement of the defined identifier.
#include	Informs the High-performance Embedded Workshop that the specified file is dependent on the current source file.
#if , #else, and #endif #elif	When #if <expression> is entered, the High-performance Embedded Workshop only analyzes the range of source code that satisfies the expression. When the value resulted from the expression is 0: #if to #else When the value resulted from the expression is 1: #else to #endif
defind macro	Entering defined (<identifier>) returns 1 when the identifier has already been defined. Otherwise 0 is returned. This macro is usually written as #if defined (MACRO).
#if def #ifndef	The High-performance Embedded Workshop assumes an #ifdef <identifier> statement as #if 1 when the identifier has already been defined. Otherwise #if 0 is assumed. An #ifndef <identifier> statement, on the other hand, is assumed as #if 0 when the identifier has already been defined, and otherwise #if 1.
#line	Ignored
#error	Ignored
#pragma	Ignored
#	Ignored

When a preprocessor statement has an expression, the following ten kinds of operators are currently supported with the dependency check. The dependency search does not operate correctly if an operator other than those listed below is used.

( ), !, <, <=, >, >=, ==, !=, &&, and ||.

1.3 Navigation facilities

1.3.1 Navigation to a definition of a C function

The High-performance Embedded Workshop scans a definition of a C function in a C/C++ source file and displays a tag of the definition on the “Navigation” tab of the “Workspace” window so that double-clicking the tag will navigate you to the line of the definition in a corresponding file shown in the editor. The High-performance Embedded Workshop evaluates no preprocessor directives when it searches a definition of a C function. For example, two definitions of func() might be displayed on the “Navigation” tab of the “Workspace” window in the following code.

```
#define DEF 1
#ifdef  DEF
void func(void)
{
}
#else
int func(int a)
{
}
```

1.3.2 Default setting for "C++ Classes" of the Navigation facilities

“C++ Classes” of the navigation facilities is disabled (unchecked) at default. You can check the setting in the “Select Categories” dialog box (figure 1.1). To open the “Select Categories” dialog box, right-click on the window of the “Navigation” tab and then select “Select Categories”

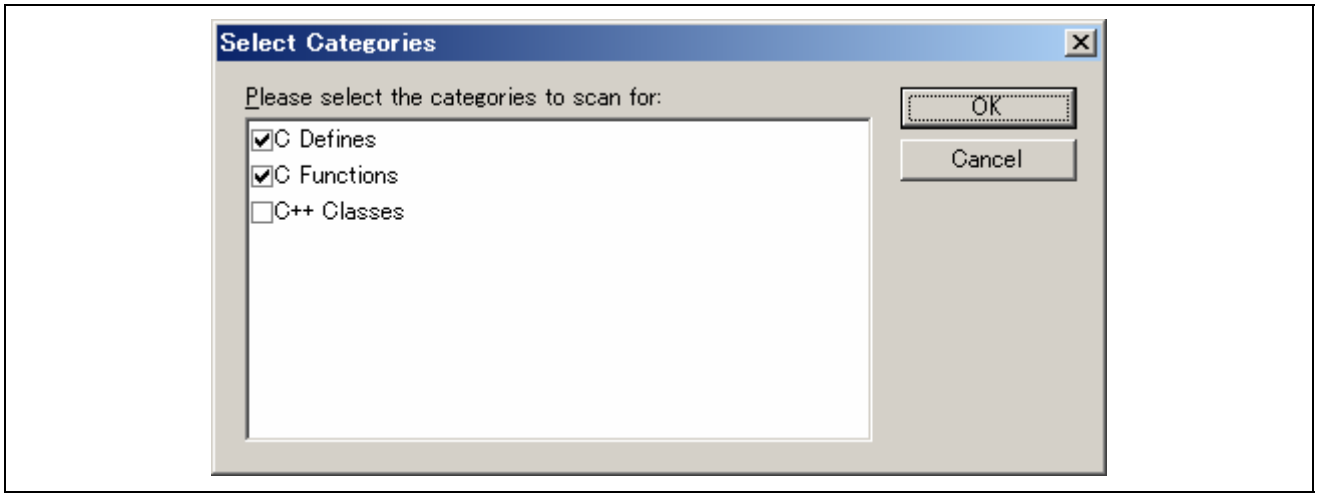

Figure 1.1 Select Categories Dialog Box

1.3.3 Warning message by "C++ Classes" of the Navigation facilities

When "C++ Classes" of the navigation facilities is enabled (checked), the warning message will appear (figure 1.2). While this item is enabled, the navigation facilities run in the background. The CPU usage of your host computer may be shown as nearly 100% (depending on the performance of the computer) and the reaction of the High-performance Embedded Workshop may be slow.

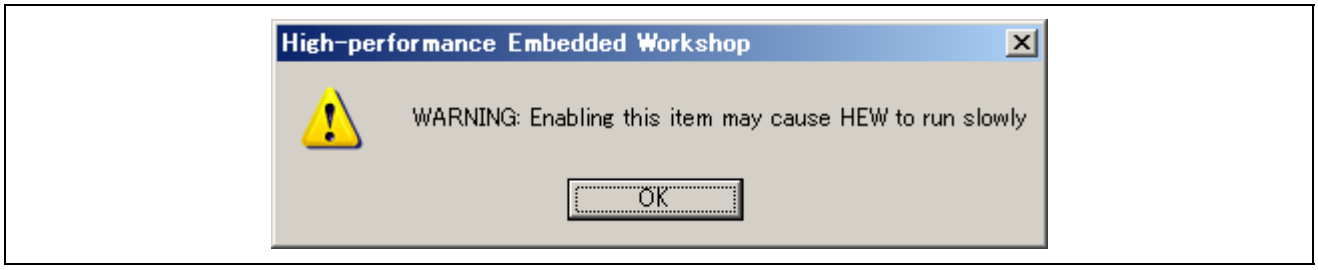

Figure 1.2 Warning Message

1.4 Expression

(1) A function name of the C++ program cannot be specified for the expression symbol.
(2) No overloaded operator can be used as a function name.

1.5 Opening a file which caused an error

This is a note when you double-click on an error/warning message of the C/C++ Compiler or the Assembler on the “Output” window and open the corresponding line of the corresponding file. If the corresponding file window, however, has been minimized in the editor window area, the file is not opened on double clicking on the error/warning message. In this case, restore the file window or maximize it.

1.6 Use of a relative path on an Option dialog box

Using the [Build] menu of the High-performance Embedded Workshop, you can launch an option dialog box on which you can specify options to a tool such as a compiler. Do NOT specify a relative path when you type a path on an option dialog box. Especially, when “Custom directory” is selected in the “Relative to” field on a dialog box as shown in figure 1.3, an absolute path must be specified in the “Directory” field.

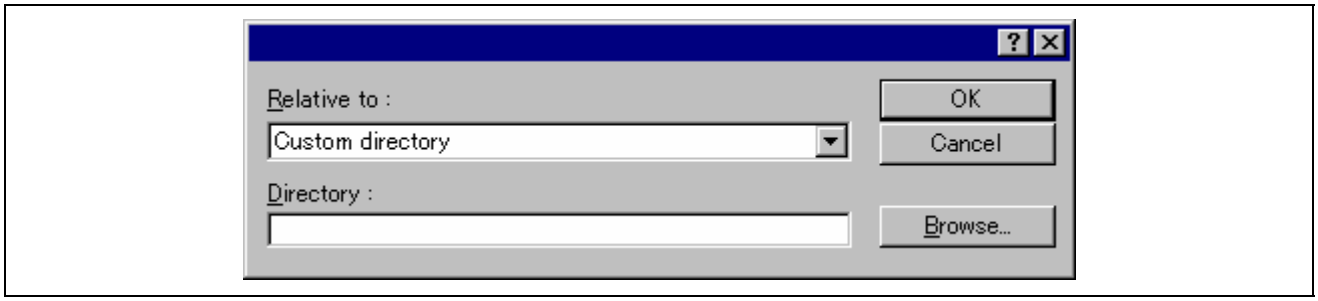

Figure 1.3 Dialog Box to Specify a Directory Path

1.7 Character sets in editor window

The editor window does not support Unicode.

1.8 Debugging (common)

1.8.1 Source-level execution

Even standard C libraries are executed by [Step In]. To return to a higher-level function, click on the [Step Out] button. In a “for” statement or a “while” statement, executing a single step does not move to the next line. To move to the next line, execute two steps.

1.8.2 Moving source file position after creating load module

When the source file is moved after the load module has been created, the [Open] dialog box, which specifies the source file, may be displayed during debugging the created load module. Select the corresponding source file and click on the [Open] button.

1.8.3 Editor window

After modifying the program displayed in the editor window and reloading the source files and load modules, close the editor window then open it again; otherwise the program displayed in the window may not be correct.

1.8.4 Disassembly window

When [Step] or [Go] is executed while the disassembly and editor windows show data in source mode and mixed mode, respectively, the program counter (PC) icon ➪ may be placed at an incorrect position in the disassembly window. If you wish to start debugging with source mode selected in the disassembly window, be sure to select source mode in the editor window.

1.8.5 Stack Trace window

When the stack trace window is opened after an interrupt function has been executed, display of data before the interrupt function is incorrect.

1.8.6 Font in Memory window

When a proportional font is selected, part of the characters in the view may be hidden. Fixed fonts are recommended.

1.8.7 Entry point

Even when an entry point is specified by the ENTRY option of the linkage editor, the entry point address is not set for the PC during program downloading. Set the PC before program execution.

1.8.8 Register window

In High-performance Embedded Workshop V.4.04 and later versions, values in registers are not saved in sessions.

1.9 Debugging (SuperH/H8 family debugger)

1.9.1 Line assembly

Regardless of the Radix setting, the default for line assembly input is decimal. Specify H’ or 0x as the radix for a hexadecimal input.

1.9.2 Loading of SYSROF format file

The debugging program of the SYSROF format cannot be loaded. Please make the debugging program by the ELF/DWARF2 format.

1.9.3 Session file of Hitachi Debugging Interface

The session file of Hitachi Debugging Interface cannot be used. Please make the file as project workspace of High-performance Embedded Workshop. The command file of Hitachi Debugging Interface can be used.

1.9.4 Profiler

Profiler does not support overlay function.

1.9.5 Watch and Locals windows

Depending on the generated object code, local variables in a C source file that is compiled with the optimization option enabled will not be displayed correctly. Check the generated object code by opening the disassembly window.

1.9.6 Configure Overlay dialog box

This dialog box may look as if a section adjacent to an overlay section is also adjacent to another overlay section.

If overlay sections are allocated as (1), (2), and (3) in example 1 and then re-allocated after build as in example 2, the [Configure Overlay] dialog box will be as shown in example 3.

Example 1

```
<Section>
(1) "P11, P12"
(2) "P21"
(3) "P31, P32"
```

Example 2

```
<Sec> <Start - End >
"P11" 0x1000 - 0x10FF (*1)
"P12" 0x1100 - 0x12FF
"P21" 0x1000 - 0x10FF (*1)
"P31" 0x1000 - 0x105F
"P32" 0x1060 - 0x11FF
```

*1. The end address of "P11" having an adjacent section is the same as that of "P21" having no adjacent section.

Example 3

```
<Section>
(1) "P11, P12"
(2) "P21, P12" (*2)
(3) "P31, P32"
```

*2. Looks as if "P12" is adjacent to "P21".

1.9.7 Simulator/debugger

(1) Memory resource setting function

Note that the specifications of the memory resource setting function of the simulator/debugger differ from those of Renesas emulators.

The specifications for the simulator/debugger are as follows:

- When the memory attribute (Read, Write, or Read/Write) is the same after the memory resource is modified:
  The simulator/debugger assumes the memory resource size has been modified and changes the memory resource size to the specified value.

- When the memory attribute (Read, Write, Read/Write) changes due to modification of the memory resource:
  The simulator/debugger assumes the memory resource attribute has been modified and changes the memory resource attribute of the specified range to the specified value.

(2) The number of breakpoints and the number of Stop-at points in the Run menu

Up to a total of 1,024 PC breakpoints and Stop-at points in the Run menu can be specified. If 1,024 breakpoints are specified, no Stop-at points in the Run menu can be specified. Specify 1,024 or fewer points for the total numbers of PC breakpoints and Stop-at points in the Run menu.

(3) Debug Settings dialog box

- Do not perform automatic target connection

The target is connected when [Debug Settings] have been completed irrespective of the setting for the [Do not perform automatic target connection] check box on the [Options] tab.

(4) SH-4 and SH-4 with BSC simulator/debugger

- In SH-4 with BSC simulator/debugger, if the lower three bits of the source and destination addresses differ in a DMA transfer, the last data transferred will be invalid.
  e.g. SAR0=2000 DAR0=4004 DMATCR0=2 CHCR0=5491
  Memory contents of address H'2000: 0102030405060708
  Contents of address H'4004 after DMA transfer: 0106
- In SH-4 and SH-4 with BSC simulator/debugger, even if an instruction is modified for an address where decoding has been completed, the pipeline is not reset and executed.
- In SH-4 with BSC simulator/debugger, if memory is accessed by a data size that differs from the size specified by the break data, the program may not break when the break conditions are satisfied. To avoid this, specify the same data size for the memory access data size and break data size.
- In SH-4 and SH-4 with BSC simulator/debugger, the pipeline execution for double precision FDIV and FSQRT instructions in the SH-4 simulator/debugger is different from those in the user system. For the SH-4 simulator/debugger, one more cycle is displayed for the F3 stage pipeline.

(5) SH-3DSP simulator/debugger

- Exception code during DSP loop execution

  If an exception is generated during the DSP loop execution, the exception code set in the EXPEVT (exception event register) will differ from that described in the programming manual.

  The exception code in the DSP Loop

General exception events	Programming Manual	Simulator
TLB miss exception/TLB invalid exception (Read)	H'070	H'040
TLB miss exception/TLB invalid exception (Write)	H'070	H'060
TLB protection exception (Read)	H'0D0	H'0A0
TLB protection exception (Write)	H'0D0	H'0C0
CPU address error (Read)	H'070	H'0E0
CPU address error (Write)	H'070	H'100

- X/Y memory access conflict

  If an instruction code and data are allocated to the XRAM memory (or XROM, or YROM, or YRAM), stalls will not be generated by a conflict even if this XRAM memory is accessed by the instruction code fetch and the MOVX or MOVY instruction in the same slot. Therefore, the number of cycles for these two types of access will differ.

- Pipeline execution from an address other than a multiple of four

  If a pipeline execution is performed from an address other than a multiple of four, the fetch stages will differ from those described in the programming manual.

  e.g. When pipeline execution is performed from an address other than a multiple of four

```
Programming Manual        Simulator
IF IF ID EX               IF ID EX
      IF ID EX               IF ID EX
         if ID EX               if ID EX
```

(6) Trace function (only for the SH3, SH3E, and SH-3DSP)

- FPU, MAC, or DSP register access information

  When an instruction is executed to write data in the FPU, MAC, or DSP register, the simulator/debugger does not display access information in the trace information.

- Pipeline display

  For instructions that programming manuals do not describe as writing back data to registers, and if data is written to registers as a result of the execution of such instructions, the simulator/debugger displays such an operation as a stage of memory access and write back in the trace information. The numbers of cycles are correctly displayed.

```
Trace display by the simulator/debugger                    Description in the
                                                           programming manual
                    IF DE EX MA SW                         IF DE EX MA SW
05  NOP             06 05                                  06 05
06  TRAPA #H'10     07 06 05                               -- 06
--  - - - - -       -- --  06                              -- -- 06
--  - - - - -       -- --  06 06                           -- -- 06
--  - - - - -       -- --  06 06                           -- -- 06
--  - - - - -       -- --  06 06 06 (06):SSR<-60000001     -- -- 06
--  - - - - -       08 -- -- 06 06 (06):SPC<-00001006      07 -- --
--  - - - - -       09 08 -- -- --                         08 07 --
```

(7) H8SX Simulator/debugger

H8SX Simulator/debugger does not support Middle mode of H8SX CPU.

(8) Selection of the simulator target

When you create a project, select the simulator target corresponding to the CPU and the operating mode.

(9) Internal RAM area in the SH2A-FPU simulator/debugger

Though the memory map of the SH2A-FPU simulator/debugger shows addresses 0xFFF80000 to 0xFFFBFFFF as the internal RAM area, the range of 0xFFFA0000 to 0xFFFBFFFF are actually reserved. For this reason, note the followings when using the SH2A-FPU simulator/debugger:

- Only use the range of 0xFFF80000 to 0xFFF9FFFF as the internal RAM area.
- When a stack area is to be assigned to the internal RAM area, set the initial stack-pointer value so that it will be within the range of 0xFFF80000 to 0xFFF9FFFF.

(10) Saving workspaces for older versions of the High-performance Embedded Workshop

In High-performance Embedded Workshop V.4.04 and later versions, workspaces can be saved in the format compatible with older versions of the High-performance Embedded Workshop. Note, however, that the operation of the simulator/debugger may be extremely slow. If you intend to use this new feature, set the trace capacity as 32,768 or less.

1.10 Windows and dialog boxes

1.10.1 Scrolling

Up-scrolling may not be available with an intelligent mouse. In this case, use a scrolling button on the window.

1.10.2 Saving the location of windows

The location of windows displayed may be changed by saving or loading of a session or switching of the virtual desktop.

1.11 Help

When the help is used in Windows® English version, the following message box may be displayed:

Click on the [Download] button to download ‘Japanese Text Display Support’, or check the [Never download any of these components.] check box and click on the [Cancel] button.

Even if ‘Japanese Text Display Support’ is not downloaded, the help is normally displayed.

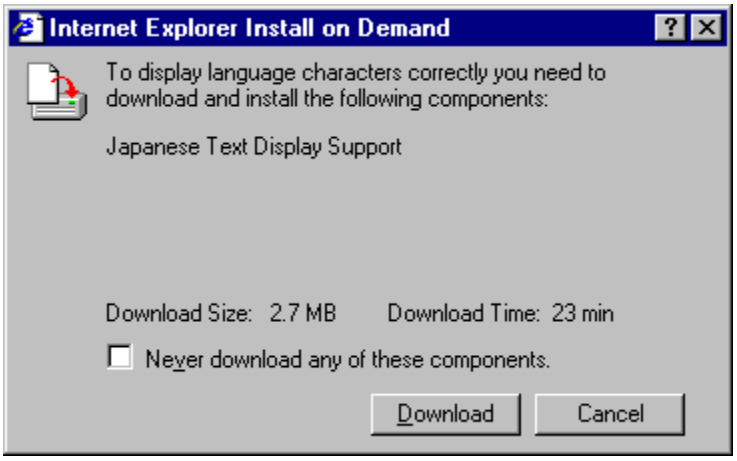

Figure 1.4: Internet Explorer Install on Demand message box

1.12 Macro-Recording Support facility

If you use the macro-recording facility to record the action of selecting [Debug -> Go] and [Debug -> Halt Program] into a macro, playing this macro will carry out [Debug -> Go] only. [Debug -> Halt Program] will not be carried out.

1.13 Commands

1.13.1 MEMORY_EDIT command

Even if characters are enclosed with quotation marks ('), the ASCII character strings cannot be input as data.
When the character strings are input as data, enter the corresponding numerical values.

1.13.2 Command batch file load timing (Debug Settings dialog box)

(1) When the [INITIALIZE] command is specified in the command file that [At target connection] has been selected for [Command batch file load timing] on the [Options] tab, do not execute the [INITIALIZE] command in the [Command Line] window.
For initialization, select [Debug -> Initialize].

(2) Do not specify the following commands in the command file that [At target connection] has been selected for [Command batch file load timing] on the [Options] tab.

(a) [OPEN_WORKSPACE] command

(b) [CHANGE_PROJECT] command

(c) [CHANGE_CONFIGURATION] command

(3) Do not specify the following commands in the command file that [Before download of modules] or [After download of modules] has been selected for [Command batch file load timing] on the [Options] tab.

(a) [OPEN_WORKSPACE] command

(b) [CHANGE_PROJECT] command

(c) [CHANGE_CONFIGURATION] command

(d) [GO] command

(e) [GO_RESET] command

(f) [GO_TILL] command

(g) [STEP] command

(h) [STEP_OUT] command

(i) [STEP_OVER] command

1.13.3 Command line batch processing (Debug Settings dialog box)

[File directory] cannot be used in the placeholder of [Command Line Batch Processing] on the [Options] tab. If a command file is specified with the relative path format, the file may not be correctly accessed.
Specify the absolute path format to the command file that cannot apply a placeholder.

1.13.4 File specification of Command Line

To specify a file in the command line, use a placeholder (excluding TCL). If you wish to specify a directory not included in the placeholder, specify an absolute path. After specifying the absolute path, this file will not be correctly found when it is in another host computer or environment where the path content is different. In such cases, specify the file again.

Example: `FILE_LOAD ELF/DWARF2 $(CONFIGDIR)\\demo.abs`

1.13.5 CACHE command

Do not specify the CACHE command unless it is defined in the user's manual of the emulator or debugger. The unit of an access to the cache memory is a fixed value such as 0x3FF. Thus the range of the cache memory being accessed may be larger than the access range specified by the user.

1.13.6 Abbreviation of a command

- REMOVE_FILE

In High-performance Embedded Workshop V.3.01 or later version, the abbreviation has been changed from "RF" to "REM".

1.14 Enhanced compatibility between TCL commands and High-performance Embedded Workshop commands

Due to enhanced compatibility between TCL commands and High-performance Embedded Workshop commands*, the execution result of a High-performance Embedded Workshop command "memory_display 300 10" will not be output by the following specification.

```
for {set i 0} {$i < 2} {incr i} {
    memory_display 300 10
}
```

To make the execution result of the High-performance Embedded Workshop command be output, enclose the High-performance Embedded Workshop command with [] as a parameter of a TCL command "puts".

```
for {set i 0} {$i < 2} {incr i} {
    puts [memory_display 300 10]
}
```

*Note: Due to this enhanced compatibility, even the execution result of a High-performance Embedded Workshop command can be assigned to a variable as a parameter of a TCL command “set”. In the example below, since the execution result of “memory_display 300 10” is assigned to variable “md_300_10”, you can view this result by specifying “md_300_10” with a TCL command “set”.

```
set md_300_10 [memory_display 300 10]
```

1.15 Tcl/Tk command input

(1) Canceling the interactive mode

To return from the interactive mode to the command input mode while using the [TCL Toolkit], enter “/.”. It is possible to know the current mode by entering “/.”.

(2) Clearing the contents of a log file

The contents of the [Console] screen are logged to a log file while using [TCL Toolkit]. The log file is in plain text format and placed in the following location:

```
C:\Documents and Settings\<logon user name>\Local Settings\Temp\log.txt
```

If you quit [TCL Toolkit], it automatically erases the contents of the log file. If you want to clear the contents of the log file while using the [TCL Toolkit], execute the following commands.

```
Set dir $env(TEMP)
set dataFile [open $dir/log.txt {RDWR TRUNC} ]
close $dataFile
```

1.16 Commands in TCL Toolkit and Command Line

(1) “trace” command

(a) To execute the “trace” command for Tcl in [TCL Toolkit], the command name must be specified in lowercase letters.

(b) To execute the “TRACE” command for Hew in [TCL Toolkit], the command name must be specified in uppercase letters.

(c) To execute the “trace” command for Tcl in the [Command Line] window, the command name must be replaced with “tcl_trace”.

(2) “clock” command

(a) To execute the “clock” command for Tcl in [TCL Toolkit], the command name must be specified in lowercase letters.

(b) To execute the “CLOCK” command for the emulator in [TCL Toolkit], the command name must be specified in uppercase letters.

(c) To execute the “clock” command for Tcl in the [Command Line] window, the command name must be replaced with “tcl_clock”.

(3) “event” command

(a) To execute the “event” command for Tk in [TCL Toolkit], the command name must be specified in lowercase letters.

(b) To execute the “EVENT” command for the emulator in [TCL Toolkit], the command name must be specified in uppercase letters.

Note that some emulators do not support commands “CLOCK” and “EVENT”.

2. Supplement on toolchain

2.1 File extensions

When the High-performance Embedded Workshop executes a tool of the toolchains in a build, the High-performance Embedded Workshop leaves a subcommand file in a configuration directory. A subcommand file of the C/C++ Compiler or the Assembler uses a file extension shown in table 2.1 with the same filename as the input file. The C/C++ Library Generator or the OptLinker uses a file extension shown in table 2.1 with a project name as a filename. The subcommand file has an attribute of a hidden file. If you want to see a hidden file, change the properties of a directory window so that it can display hidden files.

Table 2.1 File Extensions of Subcommand Files of the Toolchains

File extension	File group
shg	SuperH RISC engine C/C++ Library Generator
shc	SuperH RISC engine C/C++ Compiler
sha	SuperH RISC engine Assembler
h8g	H8S,H8/300 C/C++ Library Generator
h8c	H8S,H8/300 C/C++ Compiler
h8a	H8S,H8/300 Assembler
hlk	OptLinker
m16cl	M16C/60, M16C/30, M16C/Tiny, M16C/20, M16C/10, R8C/Tiny Linker
m16ci	M16C/60, M16C/30, M16C/Tiny, M16C/20, M16C/10, R8C/Tiny Librarian
m16cc	M16C/60, M16C/30, M16C/Tiny, M16C/20, M16C/10, R8C/Tiny C Compiler
m16ct	M16C/60, M16C/30, M16C/Tiny, M16C/20, M16C/10, R8C/Tiny mkmrtbl (MR30)
m32cl	M32C/80, M16C/80, M16C/70 Linker
m32ci	M32C/80, M16C/80, M16C/70 Librarian
m32cc	M32C/80, M16C/80, M16C/70 C Compiler
m32ct	M32C/80, M16C/80, M16C/70 mr308tbl (MR308)
m32rl	M32R C Compiler
m32ri	M32R Librarian
m32rm	M32R Load module converter
m32rc	M32R C Compiler
m32ra	M32R Assembler
741k	740 Linker
741b	740 Librarian
100l	R32C/100 Linker
100i	R32C/100 Librarian
100c	R32C/100 C Compiler
100t	R32C/100 mr100tbl (MR 100)

3. Upgrading the toolchain

When any of the following toolchains is used in a project, you can install the latest version to upgrade the toolchain currently in use.

- SuperH family C/C++ compiler package V. 5.1 or later
- H8SX, H8S, H8 family C/C++ compiler package V. 3.0A or later
- M16C series C compiler package V.5.20 release 01 or later
- M32C series C compiler package V.5.20 release 01 or later
- R32C series C compiler package V.1.01 release 00 or later
- M32R series C/C++ compiler package V.4.20 release 01 or later
- 740 family C compiler package V.1.01 release 01 or later

(1) Upgrading from the SuperH Family C/C++ Compiler Package V.5.1x or H8SX, H8S, H8 Family C/C++ Compiler Package V.3.0x

(a) Setting the standard library configuration tool

A standard library configuration tool was added. The High-performance Embedded Workshop checks the optional information on the IM Optlnker at upgrade. When there is a specification to enter the standard library, an option that generates a library is set (Mode -> Build a library file), and when there is no specification, an option that does not generate a library is set (Mode -> Do not add a library file).

(b) Setting the optimizing linker

The IM Optlnker, Librarian, and Stype Converter were integrated and became OptLinker. According to the registered states of each tool before upgrade, the optional setting of OptLinker differs after upgrade. Table 3.1 shows the succession of the option at upgrade.

Table 3.1 Succession of Option at Upgrade

No.	Before Upgrade				After Upgrade		
	IM OptLinker state	Librarian State	Stype Converter state	OptInker state	IM OptLinker optional information	Librarian Optional information	Stype Converter optional information
1	—	—	—	—	X	X	X
2	—	—	□	□	X	X	O
3	—	—	■	■	X	X	O
4	—	□	—	□	X	O	X
5	—	□	□	□	X	O	X
6	—	□	■	■	X	X	O
7	—	■	—	■	X	O	X
8	—	■	□	■	X	O	X
9	—	■	■	■	X	O	X
10	□	—	—	□	O	X	X
11	□	—	□	□	O	X	O
12	□	—	■	■	X	X	O
13	□	□	—	□	O	X	X
14	□	□	□	□	O	X	O
15	□	□	■	■	X	X	O
16	□	■	—	■	X	O	X
17	□	■	□	■	X	O	X
18	□	■	■	■	X	O	X
19	■	—	—	■	O	X	X
20	■	—	□	■	O	X	X
21	■	—	■	■	O	X	O
22	■	□	—	■	O	X	X
23	■	□	□	■	O	X	X
24	■	□	■	■	O	X	O
25	■	■	—	■	O	X	X
26	■	■	□	■	O	X	X
27	■	■	■	■	O	X	O

—: Unregistered, □: Registration not checked, ■: Registration checked, O: Succeeded, X: Not succeeded

(2) Upgrading from the SuperH Family C/C++ Compiler Package

(a) Automatic Build for optimizing MAP

In High-performance Embedded Workshop V.2.00 (SH Ver.7.0B/Ver.7.0.01/Ver.7.0.02), to realize optimization by using the external symbol allocation information that has been output by the optimizing linkage editor in the C compiler, a custom phase has been provided for rebuild. In High-performance Embedded Workshop V.2.01 or later version, automatic rebuild is enabled when MAP optimization is executed. Since the custom phase for rebuild is not required, open the Build Phases dialog box and remove a check for the 'Map optimize' phase, as shown in figure 3.1.

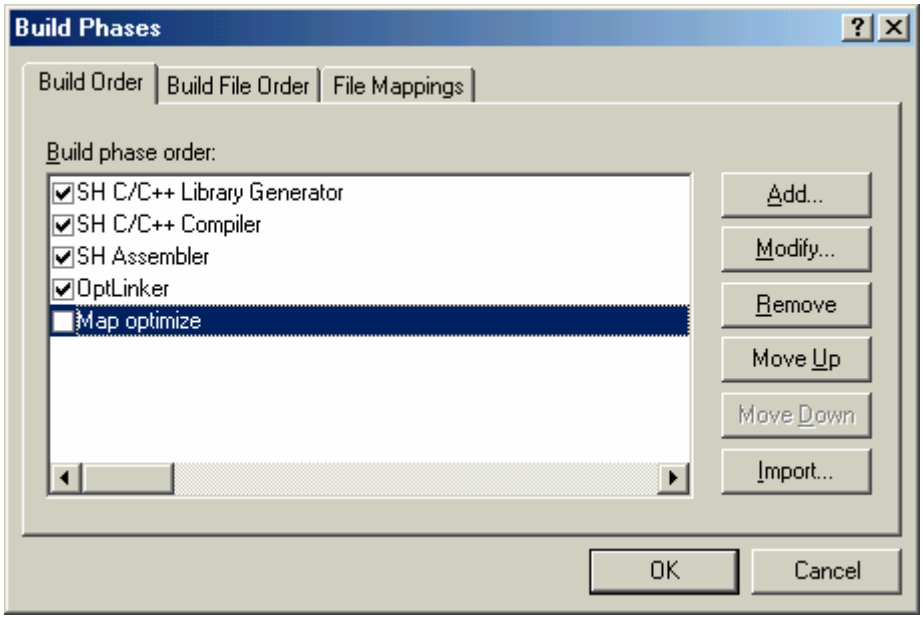

Figure 3.1 Build Phases Dialog Box